# Efficient Algorithms for Dynamic Probabilistic Safety Assessment. An Application to Convex Damage Domain

**by**

**Nicolas De los Santos Cicutto**

***THESIS***

*for the degree of*

***Master of Advanced Computing for Science and Engineering***


*Facultad de Informatica*
*Universidad Politecnica de Madrid*

*September 2011*

# Contents

# 1 Introduction

The design of nuclear power plant has to follow a number of regulations aimed at limiting the risks inherent in this type of installation. The goal is to prevent and to limit the consequences of any possible incident that might threaten the public or the environment. To verify that the safety requirements are met a safety assessment process is followed.

Safety analysis is as key component of a safety assessment, which incorporates both probabilistic and deterministic approaches. The deterministic approach attempts to ensure that the various situations, and in particular accidents, that are considered to be plausible, have been taken into account, and that the monitoring systems and engineered safety and safeguard systems will be capable of ensuring the safety goals. On the other hand, probabilistic safety analysis tries to demonstrate that the safety requirements are met for potential accidents both within and beyond the design basis, thus identifying vulnerabilities not necessarily accessible through deterministic safety analysis alone.

Probabilistic safety assessment (PSA) methodology is widely used in the nuclear industry and is especially effective in comprehensive assessment of the measures needed to prevent accidents with small probability but severe consequences. Still, the trend towards a risk informed regulation (RIR) demanded a more extended use of risk assessment techniques with a significant need to further extend PSA's scope and quality. Here is where the theory of stimulated dynamics (TSD [7]) intervenes, as it is the mathematical foundation of the integrated safety assessment (ISA) methodology developed by the CSN(Consejo de Seguridad Nuclear) branch of Modelling and Simulation (MOSI) [6, 15, 16]. Such methodology attempts to extend classical PSA including accident dynamic analysis, an assessment of the damage associated to the transients and a computation of the damage frequency.

The application of this ISA methodology requires a computational framework called SCAIS (Simulation Code System for Integrated Safety Assessment). SCAIS [2, 9] provides accident dynamic analysis support through simulation of nuclear accident sequences and operating procedures. Furthermore, it includes probabilistic quantification of fault trees and sequences; and integration and statistic treatment of risk metrics.

SCAIS comprehensively implies an intensive use of code coupling techniques to join typical thermal hydraulic analysis, severe accident and probability calculation codes. The integration of accident simulation in the risk assessment process and thus requiring the use of complex nuclear plant models is what makes it so powerful, yet at the cost of an enormous increase in complexity. As the complexity of the process is primarily focused on such accident simulation codes, the question of whether it is possible to reduce the number of required simulation arises, which will be the focus of the present work.

This document presents the work done on the investigation of more efficient techniques applied to the process of risk assessment inside the mentioned ISA methodology. Therefore such techniques will have the primary goal of decreasing the number of simulation needed for an adequate estimation of the damage probability. As the methodology and tools are relatively recent, there is not much work done inside this line of investigation, making it a quite difficult but necessary task, and because of time limitations the scope of the work had to be reduced. Therefore, some assumptions were made to work in simplified scenarios best suited for an initial approximation to the problem.

The following section tries to explain in detail the process followed to design and test the developed techniques. Then, the next section introduces the general concepts and formulae of the TSD theory which are at the core of the risk assessment process. Afterwards a description of the simulation framework requirements and design is given. Followed by an introduction to the developed techniques, giving full detail of its mathematical background and its procedures. Later, the test case used is described and result from the application of the techniques is shown. Finally the conclusions are presented and future lines of work are exposed.

## 2 Theoretical Background: Theory of Stimulated Dynamics

TSD is a variant of the stimulus driven theory of probabilistic dynamics (SDTPD) [11, 12], best suited for implementation of computer algorithms in the line of those in the classical safety assessments. The aim of these algorithms is to compute the exceedance frequency by means of combining the deterministic evolution of the plant process variables (given by a dynamic code) with stochastic transition between system configurations. Such configurations define different states, and transitions between those states are considered to follow the Markov property, thus the evolution of the system is considered as a Markov chain.

This Markov chain has the peculiarity that the transitions between states depend on the states of certain stimulus. A stimulus is either an order for an action, or a situation where the conditions necessary for a stochastic phenomenon have been met. Stimulus allow that, after some time delay, an event occurs changing the state of the system. The events mentioned early, are often related to interventions of protection systems. Also, they are usually called dynamic events as they occurrence usually provokes a change in the behaviour of the process variables (or dynamic variables).

The current system configuration and dynamic state can be described by a binary vector $\vec{i}$, such that the kth position is 1 when the kth system is active and 0 otherwise. Likewise a binary vector $\vec{I}$ can be used to specify the state of all stimulus. Thus the complete system state is denoted by the pair of binary vectors $(\vec{i}, \vec{I})$.

Figure 1 shows an example of the transitions of dynamic and stimulus state vectors where at time $\tau$ an stimulus F is activated, modifying the $\vec{I}$ vector and now allowing protective action number two to intervene. Afterwards at time $t_F$, after a delay of $t_F - \tau$, such dynamic event occurs.

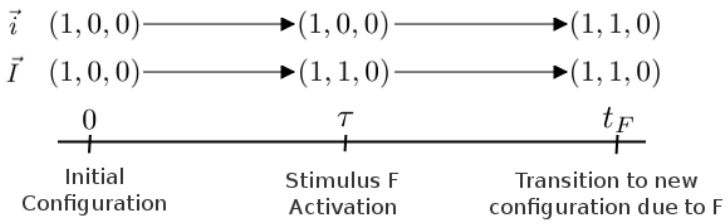



Figure 1: Stimulus activation and delay example

Let us consider a stimulus F, with an stochastic activation time delay from entered configuration $\vec{i}$ with a probability density function (pdf) $f^F_{\vec{i}}(\tau, \vec{u}(\tau))$, where $\vec{u}(t)$ is the set of process variables at time $t$ and $\tau$ the activation time. Considering that the differential probability of the delay from the activation of the stimulus to the time of the dynamic event is given by the pdf $h^F_{\vec{i}}(t - \tau, \vec{u}(\tau))$. Then the differential probability that the stimulus F provokes a change of state at time exactly t given that at time $t_0$ was at state $\vec{i}$ is

$$q_{\vec{i}\vec{j}}(t, t_0, \vec{u}) = \prod_{G \neq F, \tau_G < t} \left(1 - H^G_{\vec{i}}(t - \tau_G, \vec{u})\right) \int_0^t f^F_{\vec{i}}(\tau, \vec{u}(\tau)) h^F_{\vec{i}}(t - \tau, \vec{u}(\tau))\, d\tau \tag{1}$$

where $\tau_G$ is the activation time of stimulus G and $H^G_{\vec{i}}(t, \vec{u})$ is the cumulative density function (cdf) of the delay from stimulus activation until the dynamic event.

The previous works under the assumption that to get from the initial configuration $\vec{i}$ to some other $\vec{j}$ there is only one stimulus F that triggers the one-step transition, this restriction can be removed, but just to make things simple the assumption will be held.

As seen in reference [7] the approach to calculate the damage probability is to consider paths and sequences. A sequence is an ordered set of states or $\vec{i}$ vectors. A sequence specifies the order in which the systems will intervene. In order to define a path, the time of those interventions is required, i.e. for a sequence $\vec{s} = \{\vec{j}_n\}_{n=1}^{n=N}$, and interventions times $\vec{\tau} = \{\tau_n\}_{n=1}^{n=N}$ a path is defined by the couple $(\vec{s}, \vec{\tau})$

Now, to obtain the probability of ever entering a damage state is only a matter of considering all possible paths leading to it as follow,

$$\pi(\vec{u}) = \sum_{\vec{s}} \int_{\mathbb{R}^n} \rho(\vec{s},\vec{\tau}) \prod_{k=1}^{n-1} \frac{q_{\vec{j}_k \vec{j}_{k+1}}(\tau_{k+1},\tau_k,\vec{u})}{\prod_{\tau_G<\tau_k}\left(1-H_{\vec{i}}^G(\tau_k-\tau_G,\vec{u})\right)} d\vec{\tau} \tag{2}$$

where $\rho(\vec{s},\vec{\tau})$ is a function whose value is 1 for a damage path, and 0 otherwise.

This equation can be rewritten as,

$$\pi(\vec{u}) = \sum_{\vec{s}} \int_{\mathbb{R}^n} \rho(\vec{s},\vec{\tau}) \prod_{F\in Top(\vec{s})} \int_0^{\tau_{k(F)}} f_{\vec{i}}^F(\tau_F,\vec{u}) h_{\vec{i}}^F(\tau_{k(F)}-\tau_F,\vec{u}) \prod_{G\in Top(\vec{s})^c} \left(1-H_{\vec{i}}^G(\tau_n-\tau_G,\vec{u})\right) d\vec{\tau} \tag{3}$$

where $\tau_F$ is the time of stimulus F activation, and $\tau_{k(F)}$ is the dynamic event time that F triggers. Note that $\tau_n$ is the time of the last change of state, which must be a damage state in order to count, and thus $\tau_n$ is the damage time.

Even more, if all stimulus are set point, meaning that their activation is deterministic the equation takes the following form,

$$\pi(\vec{u}) = \sum_{\vec{s}} \int_{\mathbb{R}^n} \rho(\vec{s},\vec{\tau}) \prod_{F\in Top(\vec{s})} h_{\vec{i}}^F(\tau_{k(F)}-\tau_F,\vec{u}) \prod_{G\in Top(\vec{s})^c} \left(1-H_{\vec{i}}^G(\tau_n-\tau_G,\vec{u})\right) d\vec{\tau} \tag{4}$$

This last form of the equation will be the one used in the following sections as stimulus are considered set points with the intention of simplifying the coding of the simulation framework that will be needed.

To finalize the introduction to the TSD theory, a few comments on damage domain are added. The damage domain is defined in the TSD methodology, which is an implementation of the more general SMAP (Safety Margins Action Plan) framework [3]. It is defined as the set of non-deterministic parameters that completely define a simulation such that its evolution results in entering a damage state. Considering only the time parameters that define a path, then the damage domain for a particular sequence $\vec{s}$ is defined trough $\{\vec{\tau} : \rho(\vec{s},\vec{\tau}) = 1\}$.

# 3 Simulation Engine

As it has been mentioned already, the simulation engine of the ISA methodology is SCAIS. But in order to obtain results independent of SCAIS built-in features, an ad-hoc simulation engine will be implemented. Giving that the ultimate goal is to include the techniques that are going to be developed here, the programming language chosen for it is C++ as it is the one used for SCAIS development and therefore it will make it easier to import it into SCAIS platform.

First of all, let us discuss the basic classes that are part of the simulation process and description. These classes should describe simulation results, and TSD core concepts such as path, damage status and stimulus among others. After analysing the problem, the following classes were obtained:

- **Stimuli**: This class holds all relevant information of a set point stimulus, i.e. the distribution of the dynamic event's time delay; and it also includes a description string, to identify the different stimulus. The distribution is given with the pdf and cdf; and for certain techniques is was also included the cdf_inv function as optional, which is the inverse function of the cdf. In the future, the information given can be reduced only to the pdf, as the other functions can be obtained from it.
- **StimuliEvent**: It describes an activation event of an stimulus, deactivation was not considered on this first version as the examples that are going to be used do not require it. In order to describe an activation event two pieces of information are needed: one is the stimulus given by an instance of the *Stimuli* class, and the other is the time of event.

- **DynamicEvent**: It is used for dynamic event description. For such purpose, the stimulus that caused the dynamic event and the time of the event have to be given. The time of the event can be either an absolute time, or a delay.
- **Simulation**: After a particular path has been simulated, all information regarding that simulation has to be delivered to the probability calculation module. For that reason, this class gathers all simulation information required for computing the damage probability: the path that was simulated, the uncertain parameters sample used, the history of stimulus activation, if the simulation resulted in a damage path and the time that the simulation stopped due to entering the damage state. The path is given by a vector of *DynamicEvent* instances and the stimulus history is given by a vector of *StimuliEvent* instances. Also a boolean value, including whether the path given is valid or not has been included, this is only used when the times are given as absolute times and the dynamic events time are previous to the activation of its stimulus.

A class diagram for such classes can be seen in figure 2.

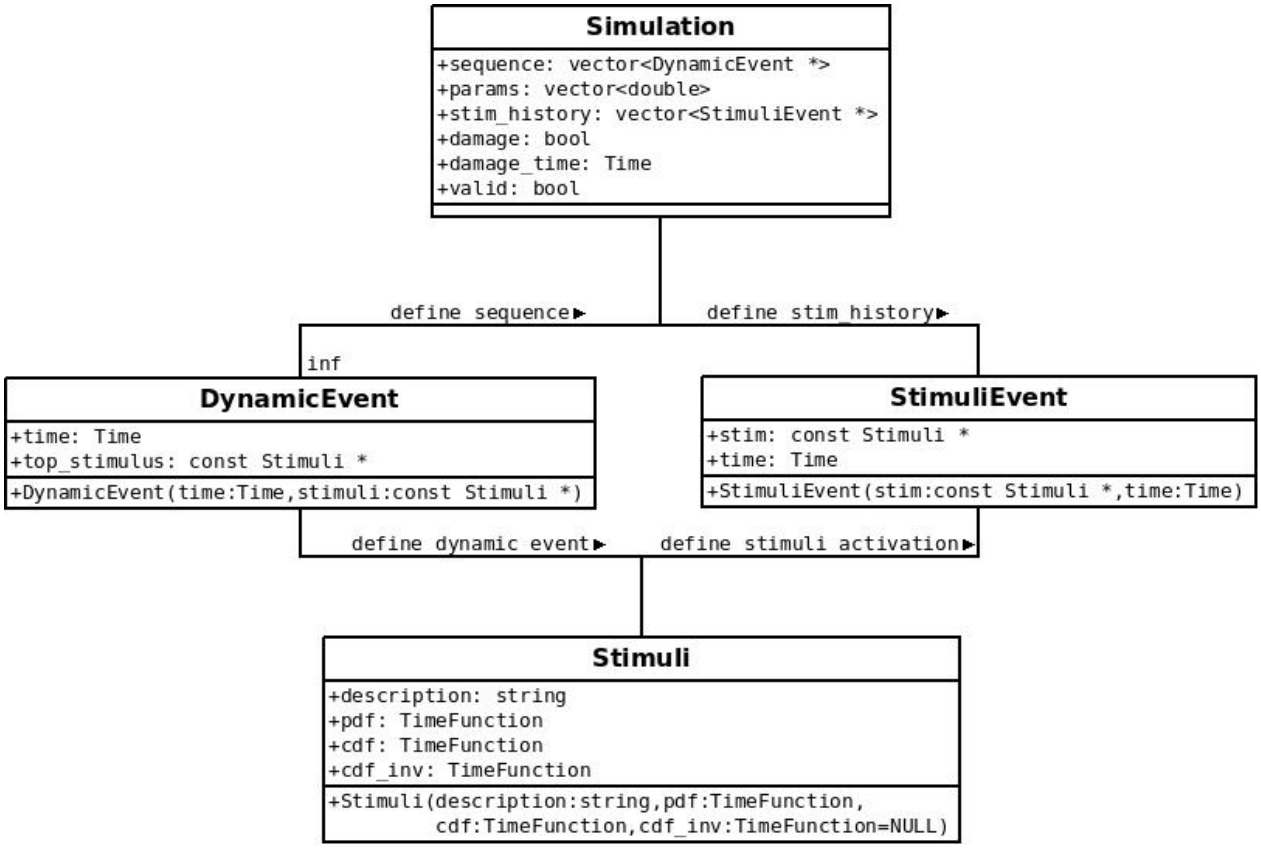



Figure 2: Simulation Classes Diagram

When commenting the *Simulation* class it was mentioned what is considered as an invalid path. In the case where a path is given to be simulated but some of the stimulus included in the path never become active, such path is not tagged as invalid as the non-activation of the stimulus only means that the path simulated is a different but valid path, and its information may be used by the probability assessment module. It is also a valid path when the stimulus is activated, but the delay is large and the dynamic event never happens. This is because it can be argued that it is a subsequence of the original sequence, which is just given by the subspace where the delays of those header which are missing in the subsequence are larger that the damage time, and thus this information is valid but for the subsequence.

The simulation classes are defined in the source files *simulation.cpp* and *simulation.hpp* that also defines

a function called simulation. This function is declared to be defined externally, as it is the function that simulates the particular dynamic behaviour for each problem, this way the simulation engine can be reused for different models. Moreover, there is also defined a vector of *Stimuli* instances called stimulus that is declared to be external too and that stores all stimulus of the specific problem. The reason for it to be external, is that it is dependent on the specific problem, and making it external maintains the generality. To sum up, the code is prepared to different dynamic codes and different stimulus definition, that should be defined in other source files containing the specifics of the problem that are to be linked at compilation time.

The simulation function should be aware of special cases that are not considered invalid path but only a simulation of a different path, and in that case the returned simulation must be include the information of the truly simulated path.

Having the simulation engine implemented, the next step is to discuss different approaches to calculate the damage probability, which is the topic of the next section.

# 4 Damage Probability Calculation Algorithms

## 4.1 Monte Carlo Simple

Let us consider a random vector $\vec{S} = (\vec{p}; \vec{\tau})$ including all random parameters that define a simulation, where $\vec{\tau}$ are all possible time delays and $\vec{p}$ are other kind of parameters. Consider now $X$ a random variable such that,

$$X(\vec{S}) = \begin{cases} 1 & \text{when the simulation of } \vec{S} \text{ gives a damage path} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Remember that simulation of $\vec{S}$ produces all possible sequences, as non-stimulated events and events with large delays are ignored by the simulation engine.

$X$ has a Bernoulli distribution. An the $P(X = 1)$ gives the damage probability. Knowing that the $E(X) = P(X = 1)$, then the mean and the damage probability can be estimated with the following unbiased estimator

$$E(X = 1) = P(X = 1) \simeq G_N = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{6}$$

Where $X_i$ are independent random variables with X distribution.

These estimation is better as N grows since $Var(G_N) = \frac{1}{N} Var(X) = \frac{1}{N} P(X = 1)(1 - P(X = 1))$. Substituting in this equation the estimation of the damage probability gives an estimation of the variance.

$$Var(G_N) \simeq \frac{1}{N} G_N (1 - G_N) \tag{7}$$

Using Central Limit Theorem (CLT) approximating $G_N$ with a normal distribution a confidence interval for the estimation can be given by the implemented code.

Using these, an algorithm for computing the damage probability can be crafted. A description for such algorithm can be found on algorithm 1.

```
for  i = 1 to N do
    S⃗ ← Sample Simulation Parameters
    sim ← Simulate(S⃗)
    if state(sim) is damage then
        damage_counter++
    end
end
return damage_counter/N
```

**Algorithm 1**: Monte Carlo Simple

The source code for this technique can be found in file *mc_ explorer.cpp*. It coded in C++ and uses the gsl library for probability and statistical uses.

## 4.2 Monte Carlo Conditional

This algorithm is based on reference [10], where is proposed as a new technique to increase efficiency of the probability calculation process.

The basic idea is to consider the probability of reaching a damage state conditioned by the first branching made. So, the first step is to simulate the root branch, the one where no intervention is made, and look at the stimulus activated. Those stimulus give all the possible branching points from the root branch. If $\tau_1 < \tau_2 < .. < \tau_n$ are the times when these $n$ stimulus are activated and $\tau_{n+1}$ is when damage occurs, then the damage probability can be express by,

$$\pi(\vec{u}) = \sum_{i=1}^{n} \int_{\tau_i}^{\tau_{i+1}} \pi(\vec{u}|\bar{\tau}_i = \tau) \sum_{j=1}^{i} h^i(\tau - \tau_j, \vec{u}) \prod_{k!=j, k\leq i} \left(1 - H^k(\tau - \tau_k, \vec{u})\right) d\tau \tag{8}$$

where $\bar{\tau}_i$ is the dynamic event time associated to stimulus $i$ and $\pi(\vec{u}|\bar{\tau}_i = \tau)$ is the conditioning probability of reaching a damage state when the event $i$ takes place at time $\tau$ and no other event can take place before that time. As before $h(\tau, \vec{u})$ and $H(\tau, \vec{u})$ are the pdf of the delays.

The previous equation can be rewritten as

$$\pi(\vec{u}) = \sum_{i=1}^{n} \sum_{j=1}^{i} I_{ij}(\vec{u}) \tag{9}$$

where

$$I_{ij}(\vec{u}) = \int_{\tau_i}^{\tau_{i+1}} \pi(\vec{u}|\bar{\tau}_i = \tau) h^i(\tau - \tau_j, \vec{u}) \prod_{k!=j, k\leq i} \left(1 - H^k(\tau - \tau_k, \vec{u})\right) d\tau \tag{10}$$

To estimate the conditioning probability a Monte Carlo game can be used. And to compute the integral in this first version of the algorithm, a Monte Carlo estimation is going to be used. For that purpose let us assume it is possible to rewrite $I_{ij}$ as the integral of a pdf multiplied by a weight function then, using samples of the distribution given by that pdf, an estimation of the integral could be

$$I_{ij} \simeq \bar{I}_{ij} = \frac{1}{N} \sum_{k=1}^{N} w_{ij}(s_{ij}^k) \tag{11}$$

where $s_{ij}^k$ are the samples, and $w_{ij}(t)$ is the weight function.

The simplest form of this estimator is to consider uniform sampling, i.e. the sampling pdf is $\gamma_{ij}(t) = 1/(\tau_{i+1} - \tau_i)$ and weight function the integrand multiplied by the factor $(\tau_{i+1} - \tau_i)$. This is the version implemented in the source file *mcc_ explorer.cpp*, but it is prepared for future changes in the weight function and pdf.

The integration over the blockage fraction parameter, as in many of the following strategies is going to be made using the trapezoidal rule. A pseudo-code of the algorithm can be found at algorithm 2

```
sim ← Simulate(Root)
foreach τ_i, τ_{i+1} do
    for j = 1 to i do
        num_samples ← 0
        while more samples do
            τ ← sample_tau_pdf(i,j)
            cond_pi ← calculate_conditioning_probability(τ)
            weight =← statistical_weight(i,j,τ)
            sum ← sum + cond_pi*weight
            num_samples ← num_samples + 1
        end
        integral ← integral + sum/num_samples
    end
end
return integral
```

**Algorithm 2**: Monte Carlo Conditional

## 4.3 Monte Carlo Conditional Version 2

This is a version of the Monte Carlo Conditional algorithm where the integral is computed by one dimensional numerical integration schemes. For that reason the gsl library integration functions are going to be used, in particular the *gsl_ integration_ qags* function. This function provides a numerical integration method based on quadrature and with an adaptive integrator that works with given precision.

The source code can be found in file *mcc_ explorer_ v2.cpp*.

## 4.4 Repeated One-dimensional

Looking at equation 4, the only thing that is needed for computing the damage probability is to solve a multidimensional integral. In order to do so, a repeated one-dimensional integration scheme that takes advantage of one dimension integration methods recursively can be used.

To take fully advantage of the repeated one-dimensional scheme, it has to be considered which are the possible branching sequences at each point in the recursion. In fact using these approach would be equivalent to considering equation 8, and having a recursive procedure compute the conditioning probability $\pi(\vec{u}|\bar{\tau}_i = \tau)$. Computing such probability is equivalent to integrating over the other dimensions and at the same time, it can be computed using the same procedure.

Then the algorithm for the recursive method should be similar to what was done in the case of Monte Carlo Conditional case. This means that such routine would require a history of dynamic events and a current time, and use it to recursively compute the conditioning probability of ever entering a damage state given that history. In order to do so, it must simulate the base branch without further intervention, and study which stimulus are capable of provoking a branching point, ahead of the current time, which are in fact knew dimensions to integrate on.

An algorithm describing these procedure can be seen in algorithm 3, where *r1d* stands for a routine that evaluates to the value of the integrand, used by the one dimensional integration method, and that at the same time will need to call to this routine again for evaluating the conditional probability until no more branching is possible.

**Data**: Base Sequence: $\vec{S}$
sim $\leftarrow$ Simulate($\vec{S}$)
**if** *state(sim) is damage* **then**
  dam_prob $\leftarrow$ compute_sim_probability(sim)
  **foreach** *stimulus able to provoke dynamic event* **do**
    dam_prob $\leftarrow$ dam_prob+ one_dimension_integration($\tau$, $t_{damage}$, stimulus, r1d)
  **end**
**end**
**return** dam_prob

**Algorithm 3**: Repeated One-dimensional

File *r1d_ explorer.cpp* stores the source code for this algorithm.

## 4.5 Repeated One-dimensional Version 2

This techniques is based on Repeated One-dimensional algorithm with the difference that the integration space is not the time domain. A variable change in equation 4 has been made in an attempt to reduce errors in the numerical integration

Assume $\alpha_k = H^k(\tau_k)$, then $d\alpha_k = h^k(\tau_k)\, d\tau_k$, and the equation can be rewritten to

$$\pi(\vec{u}) = \sum_{\vec{s}} \int_{[0,1]^n} \rho(\vec{s}, \vec{H}^{-1}(\vec{\alpha})) \prod_{G \in Top(\vec{s})^c} \left(1 - H_{\vec{i}}^G(t_{damage} - \tau_G, \vec{u})\right) d\vec{\alpha} \tag{12}$$

This change, transform the integration space, from times to probabilities. Making this change will hopefully have an impact on the precision as the distance used in now based in the probability.

Source code for this algorithm can be found at file *r1d_ explorer_ v2.cpp*

## 4.6 Damage Time Interpolation Scheme

This technique is based on the observation that in equation 12 the damage time delay affects only $H_{\vec{i}}^G$ function, and this function's image domain is inside $[0, 1]$ interval. Then it may be possible to use instead of the damage time delay, an approximation to it, and the error made in the interpolation would be expected to have a low impact on the output of the function as the results of its application is to down scale the input.

Therefore, to reduce the number of simulation required to obtain a good approximation of the damage probability, an idea would be to select a few paths, and interpolate the obtained damage delays, to be later used in the integration method. Still there is a need to estimate whether the path are damage or not, here is where the truly protection system hypothesis comes in. Truly protective systems are those that considering all intervention times fixed except one of them, increasing that time produces the damage time to increase or stay the same; and once entered the damage state, increasing the intervention time can not result in a damage path. Combined all this, means that the damage domain is a convex set, and using this information it can be determined the path status without simulation.

Moreover, since the damage domain is suppose to be convex, an efficient way to determine the damage domain boundaries, once all times except one are fixed, should be used. For that reason a binary search kind of method, where a number of simulations is given, which tries to determine enough points for estimating the damage domain's boundaries, and that is going to use those simulation for interpolating the damage time delay, is going to be used.

This algorithm, with this complex search mechanism, that uses damage delay interpolation for each stimulus, and that uses an special function to decide in each step the number of simulation to be used depending on the size of the interval to explore, can be found in file *dint_ explorer.cpp*. The procedure is very similar to the

second version of the Repeated One-dimensional algorithm, where the integral is obtained recursively, with the particularity of including the interpolation mechanism.

Without further details on the exact search mechanism, and the damage delay interpolation procedure or the function that gives the number of simulations required for each interval, which probably are receptive to improvement, let us continue with the test example.

# 5 Test Case

In order to study the developed techniques, an already well studied schematic example has been selected. For such example, validated damage domains and an estimation of the damage probability are supplied at refereces [4, 13] making it possible to assess the technique's efficiency and accuracy and to compare the obtained damage domains.

Moreover, the example includes one uncertain parameter, the blockage fraction, and the damage domains are convex sets, making it more suitable for the application of simple techniques which can be elaborated further in the future. The hypothesis of convex domain is not unreasonable under certain circumstances as will be seen in following sections.

## 5.1 System description

The system at study is a water tank in a cylindrical shape with a cross section $A = 1\,m^2$ and a height $H = 1\,m$. Water is provided to the tank with flow rate $Q_{in}$. At the same time the water is being drained with flow rate $Q_{out}$. In normal operation $Q_{in} = Q_{out} = Q_{ini}$ where $Q_{ini} = 0.005\,m^3/s$. Therefore the water level remains constant at the initial level $l_{ini} = 0.8\,m$.

There is a protection system that tries to avoid tank overflow in case that the drain pipe gets blocked. Such system is composed by two protective actions:

- **PA1:** The first protective action is the closure of the pneumatic feed valve for the incoming water pipe. This action causes the incoming water flow rate to be reduced at constant rate until the valve is completely closed. The closing time of the valve $t_c$ is 20 seconds.
- **PA2:** This action consists in opening a motor-drive emergency drain valve. When opened, the flow through this valve $Q_{em}$ increases at a constant rate until it reaches the maximum value $Q_{ev}^{max} = 0.004\,m^3/s$. The time needed for the motor to be fully operative is $t_{em} = 60\,s$.

A schematic of the tank system can be found at Figure 3.

The intervention of this actions is subject to the activation of two alarm signals dependent on the water level.

- High level signal: 0.95 m.
- High-high level signal: 0.98 m.

The High level signal, and the High-high level signal indicate respectively that PA1 and PA2 are allowed to be activated. Therefore there is an uncertain delay between the activation of the signal and the intervention of the system.

The accident considered is the blockage of the normal drain pipe which has an estimated frequency of $\nu_{ini} = 2\,y^{-1}$.

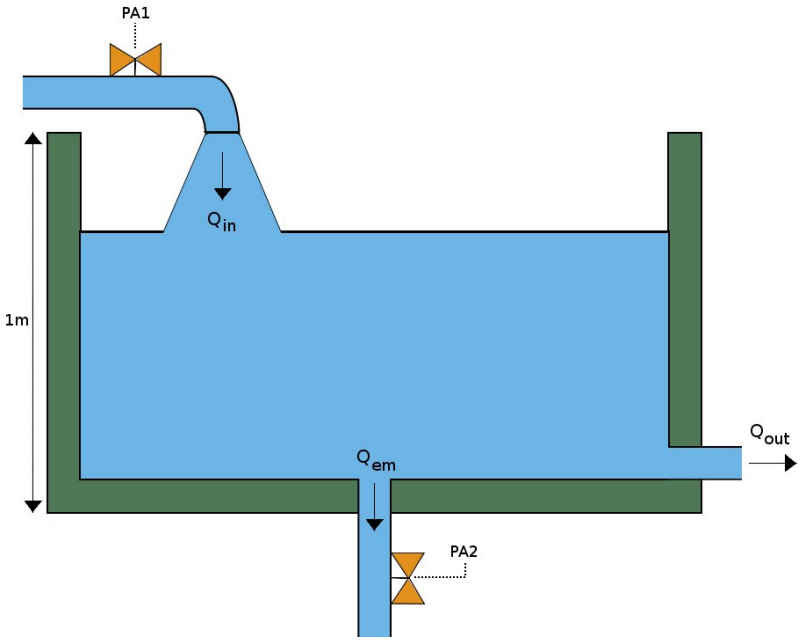



Figure 3: Tank Schema

### 5.2 Distribution of random phenomena

The drain pipe is designed to drag small objects so the blockage fraction varies from 0.5 to 1 with a probability density given by

$$f(s) = -7.2s + 7.4 \quad s \in [0.5, 1] \tag{13}$$

The delay in PA1's intervention is characterized by a lognormal distribution with parameters $\mu = 1.354$ and $\sigma = 0.8231$ (denoted as Log-$\mathcal{N}$ (1.345, 0.82312 )). This means that the *pdf* of the delay is given by

$$f(dpa_1) = \frac{1}{0.82312 dpa_1 \sqrt{2\pi}} \exp\Big( - \frac{(\ln(dpa_1) - 1.345)^2}{2 \cdot 0.82312^2} \Big) \tag{14}$$

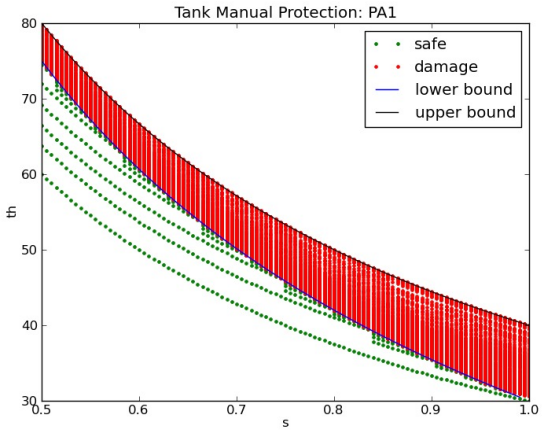
Moreover, delay in PA2's intervention is also considered a lognormal distribution $\big($Log-$\mathcal{N}$ (0.4581, 0.6999 )$\big)$, i.e. the *pdf* of the delay is given by

$$f(dpa_2) = \frac{1}{0.6999 dpa_2 \sqrt{2\pi}} \exp\Big( - \frac{(\ln(dpa_2) - 0.4581)^2}{2 \cdot 0.6999^2} \Big) \tag{15}$$

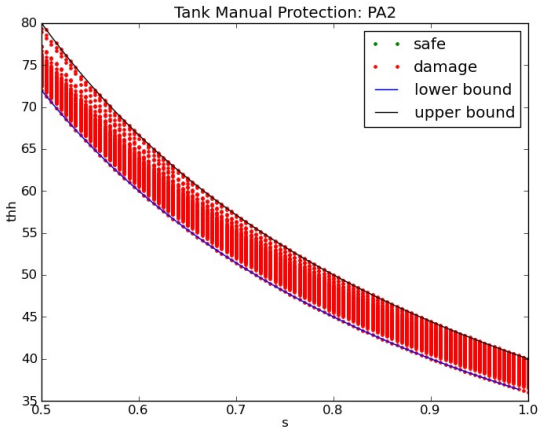
## 6 Results

### 6.1 Damage Domains

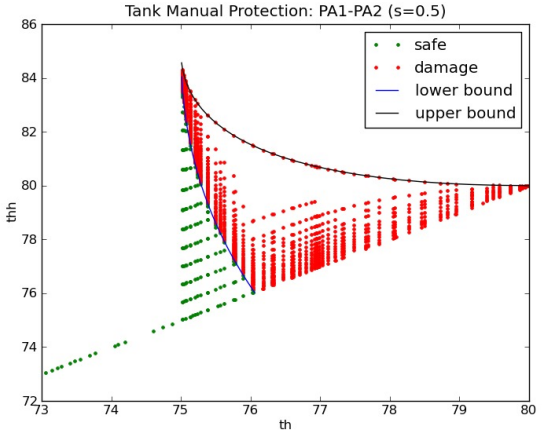
Figures 4, 5, 6, 7 and 8 show the damage domains for the algorithms: damage time interpolation, repeated one-dimensional, repeated one-dimensional version 2, monte carlo conditional and monte carlo conditional version 2, respectively. All these algorithms use the same integration method for the blockage fraction parameter, so that it were possible to obtain comparable plots and results independent of the parameter treatment.



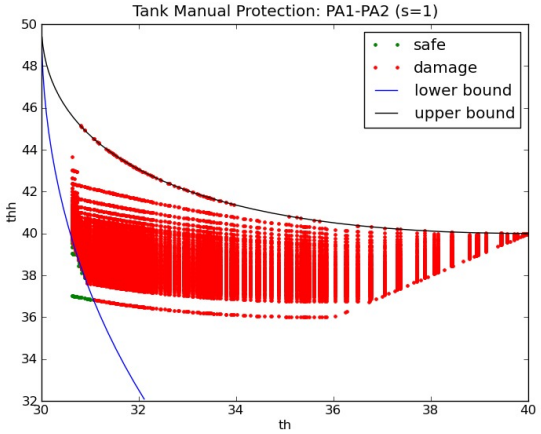

(a) PA1 sequence



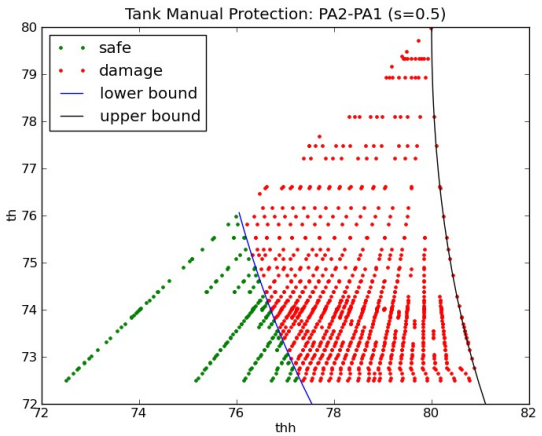

(b) PA2 sequence



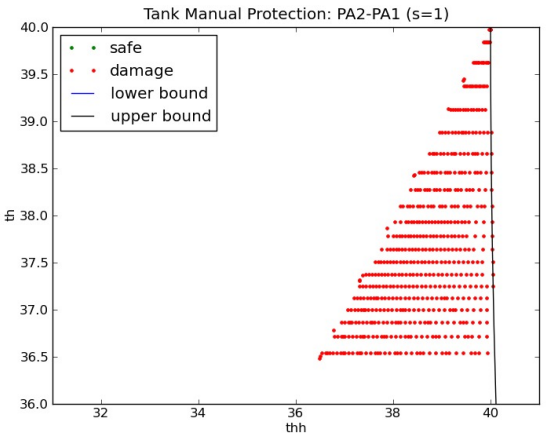

(c) PA1-PA2 sequence for s=0.5




(d) PA1-PA2 sequence for s=1.0

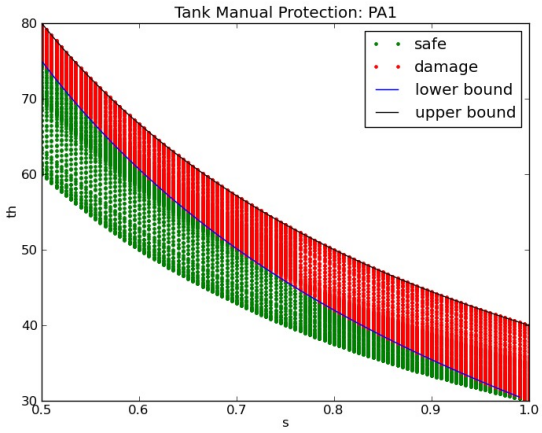



(e) PA2-PA1 sequence for s=0.5

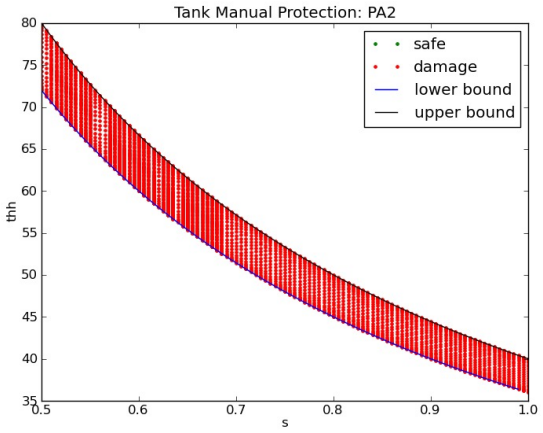



(f) PA2-PA1 sequence for s=1.0

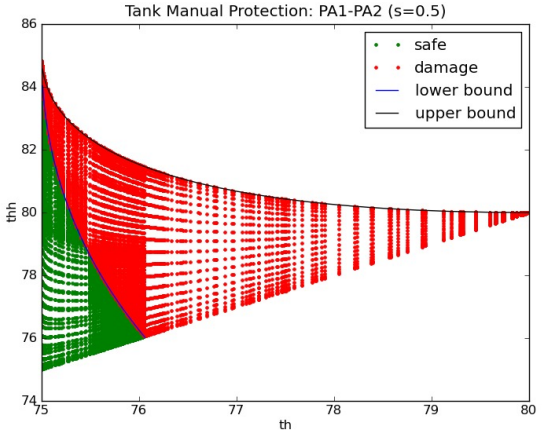
Figure 4: Damage Domains for Damage Time Interpolation Scheme Algorithm



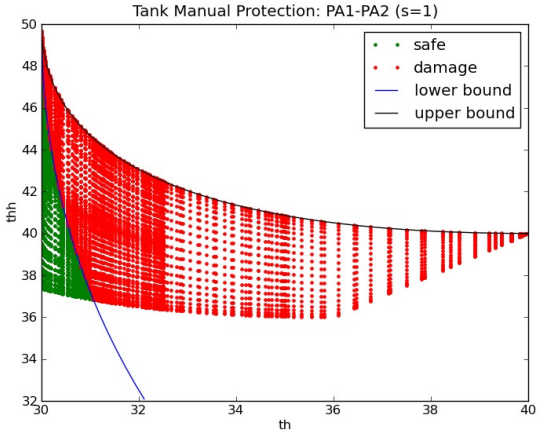

(a) PA1 sequence



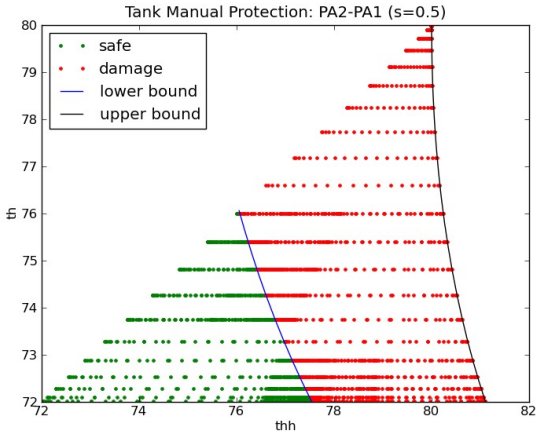

(b) PA2 sequence



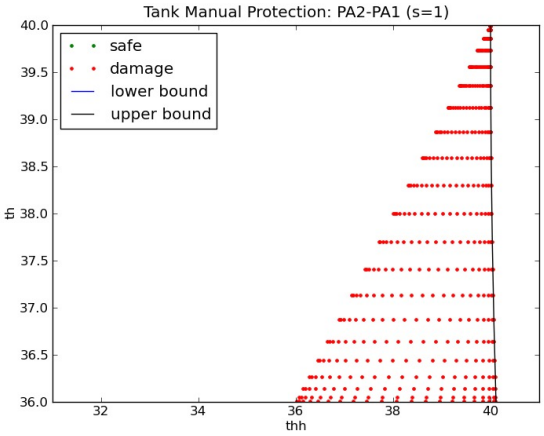

(c) PA1-PA2 sequence for s=0.5




(d) PA1-PA2 sequence for s=1.0

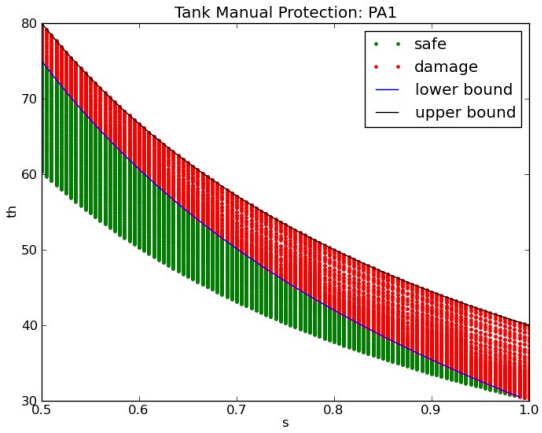



(e) PA2-PA1 sequence for s=0.5

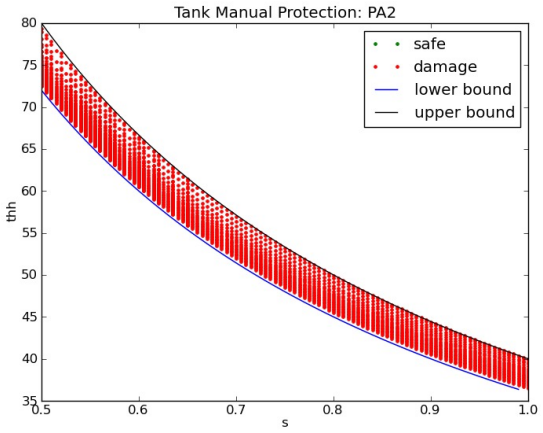



(f) PA2-PA1 sequence for s=1.0

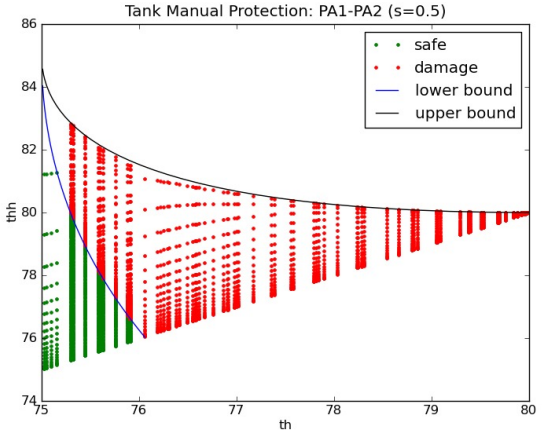
Figure 5: Damage Domains for Repeated One-dimensional Algorithm



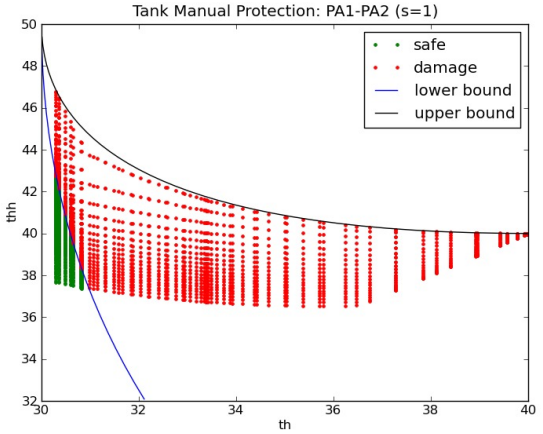

(a) PA1 sequence



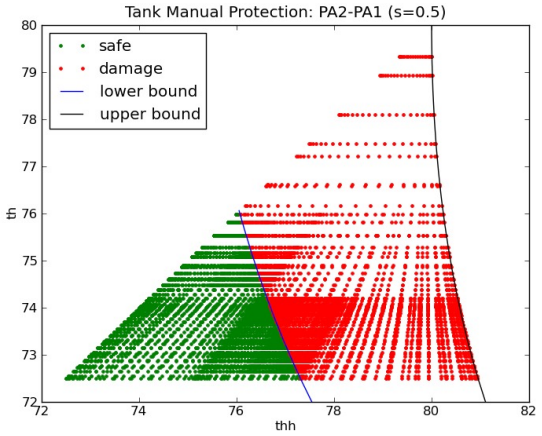

(b) PA2 sequence



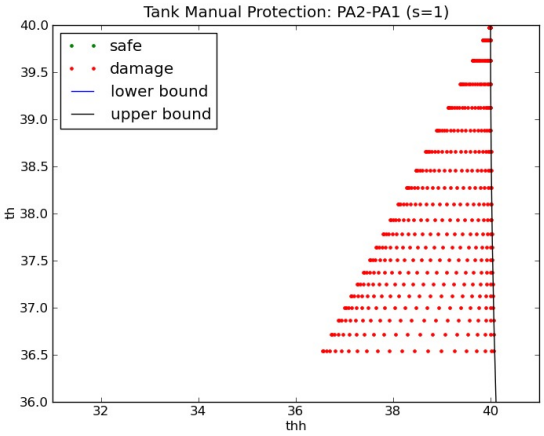

(c) PA1-PA2 sequence for s=0.5




(d) PA1-PA2 sequence for s=1.0

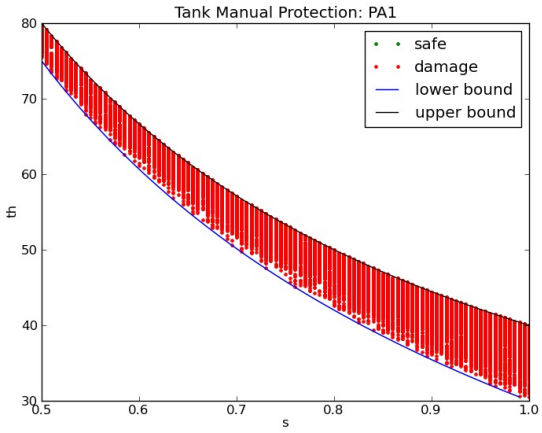



(e) PA2-PA1 sequence for s=0.5

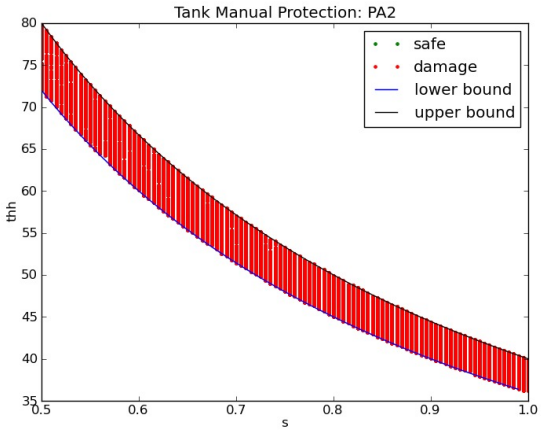



(f) PA2-PA1 sequence for s=1.0

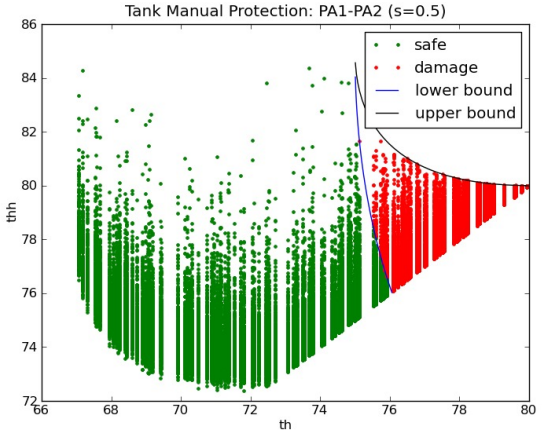
Figure 6: Damage Domains for Repeated One-dimensional Version 2 Algorithm



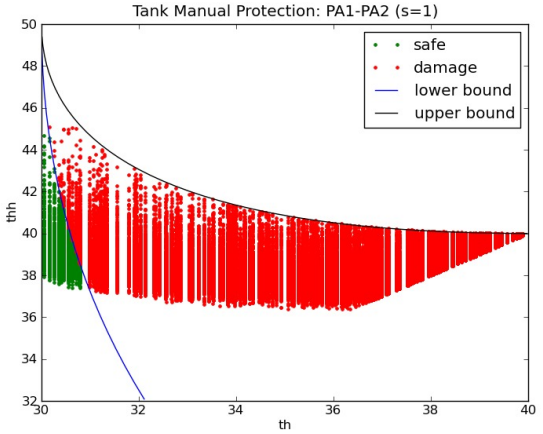

(a) PA1 sequence



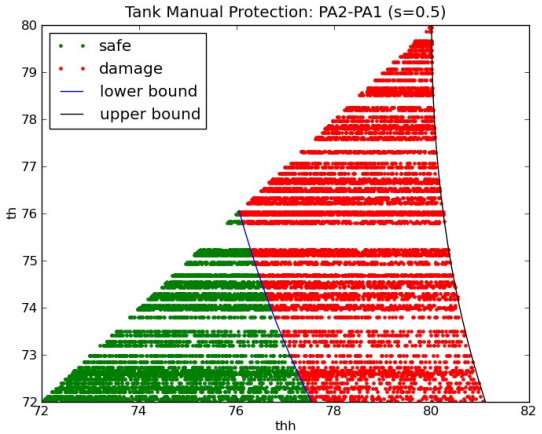

(b) PA2 sequence



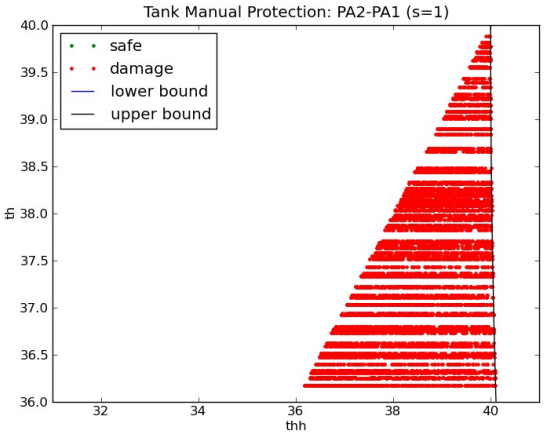

(c) PA1-PA2 sequence for s=0.5




(d) PA1-PA2 sequence for s=1.0

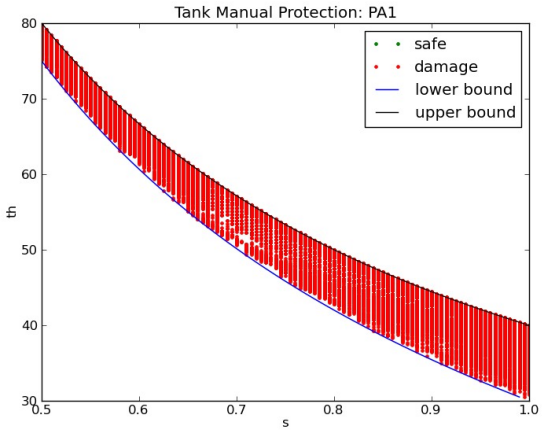



(e) PA2-PA1 sequence for s=0.5

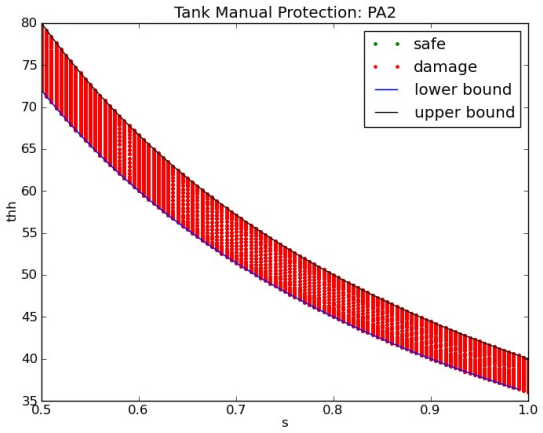



(f) PA2-PA1 sequence for s=1.0

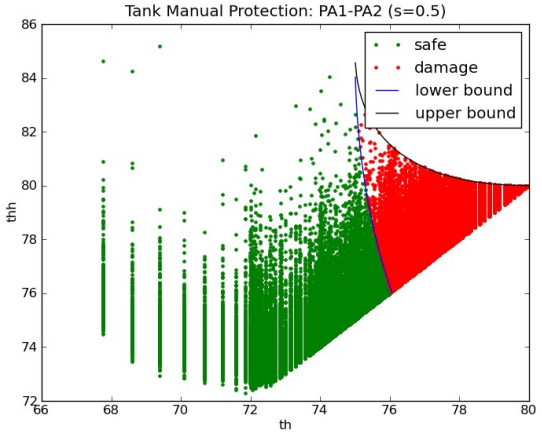
Figure 7: Damage Domains for Monte Carlo Conditional Algorithm



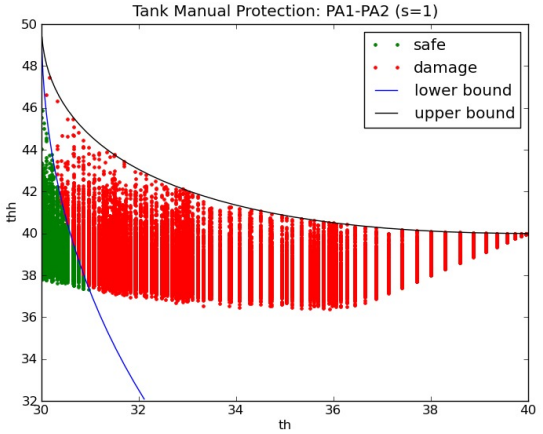

(a) PA1 sequence



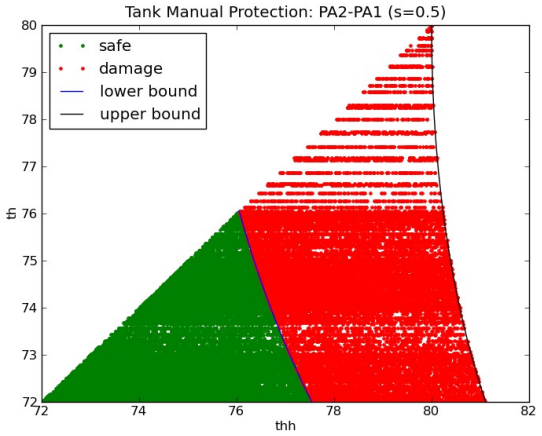

(b) PA2 sequence



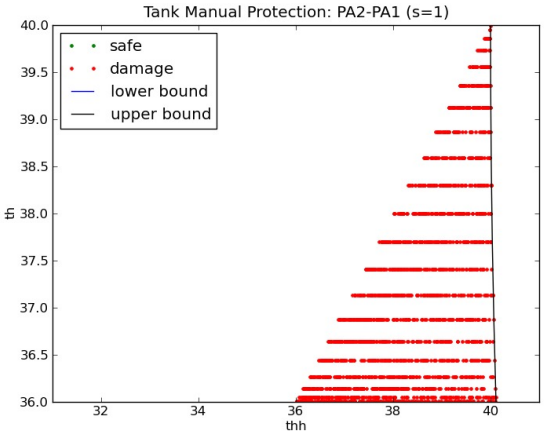

(c) PA1-PA2 sequence for s=0.5




(d) PA1-PA2 sequence for s=1.0




(e) PA2-PA1 sequence for s=0.5




(f) PA2-PA1 sequence for s=1.0

Figure 8: Damage Domains for Monte Carlo Conditional Version 2 Algorithm

## 6.2 Numerical Values

Table 1 shows the results of the application of the algorithms based on random sampling through several calls. The data gathered includes the mean damage probability estimation, the standard deviation of the samples, the mean number of simulations when the number of simulation is variable and the number of simulations when it is not, and the number of samples or calls to the algorithm. At the right of the table, in some cases appears a list of numbers, that represent the parameters set when launched the algorithm. Different results for different configurations of the algorithms are provided to compare the effect of the number of simulations in the precision and vice versa.

| Method | Mean Prob. | Std.Dev. | Mean Simulation | Samples | |
|---|---|---|---|---|---|
| Monte Carlo Simple | 0.20547774141 | 1.10 E-04 | 1.0 E+08 | 500 | |
| | 0.205474990654 | 1.04 E-04 | 1.4 E+07 | 500 | |
| | 0.20546475 | 1.44 E-04 | 8.0 E+06 | 500 | |
| Monte Carlo Conditional | 0.204611743364 | 1.52 E-03 | 13635101 | 300 | (150, 300) [1] |
| | 0.204530423077 | 1.60 E-03 | 9090101 | 300 | (100, 300) |
| Monte Carlo Conditional V2 | 0.2044745584984 | 2.98 E-05 | 11900801 | 300 | (0.175, 300)[2] |
| | 0.204475492886 | 3.08 E-05 | 9406001 | 300 | (0.15, 300) |

Table 1: Random Methods Results

Most of the algorithms use the same parameter treatment procedure, which is, using the trapezoidal rule with a step of 0.005 to integrate the conditioning probabilities over the blocking fraction parameter. This means that the algorithms have to estimate 100 damage probabilities over the interval $[0.5, 1]$, in order to compute the overall probability. For each algorithm, all 100 conditioning probability were printed. To understand how well it approximates the real conditioning probabilities, an estimation with the Monte Carlo Simple algorithm using $1E + 08$ simulation was used as reference. The reason is that the approximation obtained with such large number of simulation is probably very close to the real value.

| Method | Prob. | Sqr. Error | Abs. Error | Simulations | |
|---|---|---|---|---|---|
| Repeated One-dimensional | 0.205451834816 | 3.68 E-09 | 2.59E-05 | 1396181 | 0.0572[3] |
| | 0.20542943756 | 6.49 E-08 | 4.83E-05 | 791738 | 0.25 |
| Repeated One-dimensional V2 | 0.205449660218 | 3.30E-09 | 2.81E-05 | 1282634 | |
| Damage Time Interpolation | 0.205140702122 | 1.40E-07 | 3.37E-04 | 1400027 | (5, 100)[4] |
| | 0.204931781162 | 4.13E-07 | 5.46E-04 | 788231 | (10, 75) |
| Monte Carlo Conditional | 0.204611743364 | 4.78E-04 | 8.66E-04 | 13635101 | (150, 300) [1] |
| | 0.203405380344 | 5.56E-04 | 2.07E-03 | 9090101 | (100, 300) |
| Monte Carlo Conditional V2 | 0.2044745584984 | 9.49E-06 | 1.00E-03 | 11900801 | (0.175, 300)[2] |
| | 0.204475492886 | 9.68E-06 | 1.00E-03 | 9406001 | (0.15, 300) |

Table 2: Random Methods Results

Table 2, shows the results of comparing every blockage fraction conditioning probability with the reference values as explained previously. The table includes the damage probability, the mean square error of the conditioning probabilities, an approximation of the absolute error in the estimation of the damage probability and the number of simulations required.

As it can be seen in these tables, the least absolute error for this case comes from the repeated one-dimensional method. This means also, that is the most efficient one, i.e. with the same amount of simulations it produces a better approximation.

[1] First element is the number of samples of $\tau$, and the second one is the maximum number of simulations to do when estimating the conditional probability

[2] First element is the relative error passed to the integration method, and the second one is the maximum number of simulations to do when estimating the conditional probability

[3] relative error passed to the integration method

[4] Arguments for the number of simulation selection function

# 7 Conclusion

It has been said that this work was conceived as a first approach to studying efficient damage probability calculation methodologies that could fit inside an ISA. In order to do so, the scope of the problem was reduced to convex domains, and in particular to the water tank model. Using these model, the results obtained show that the best approach, at least for low dimension, is to use the repeated one-dimensional algorithm as it is the one that produces the most accurate results with same number of simulations. Between the two versions of the repeated one-dimensional method, the second version seems to provide better results, at least for this particular case.

The result was, at least partially, expected. The best result should have come from algorithms not based on Monte Carlo techniques, as they often require a large number of samples to obtain a good approximation. Monte Carlo techniques usually work better on higher dimension as the variance of the estimator tends to decrease as the number of samples grow, and the number of samples needed for a good approximation in other techniques usually grow exponentially with the dimension. Also notice that in low dimension as in this case, Monte Carlo Simple algorithm seems better, but it has to be considered if this still holds in higher dimensions.

The Damage Time Interpolation Scheme, could not finally compete with the repeated one-dimensional techniques. Since this techniques were explicitly making the convex assumption it was expected to have better results. But its failure probably comes not from the idea behind it but the many implementation decisions made that can, and possible will, be refined in the future. At least, in the damage domains plots, it can be seen that the damage domain is swept avoiding, as much as possible, analysis of too many safe paths.

Furthermore, even if at some point this algorithm receives good results, the decision as to when to use is still not clear. Knowing a priori if the protective systems are truly protective systems for the damage considered may be a hard task. But it may be possible to run the algorithm, checking from time to time if the hypothesis of convex set is still applicable, or even consider a convex subdivision of the damage domain and apply it on every component.

Is still early to conclude which algorithm is better, and in which conditions. But the biggest contribution of this work are the several techniques that have been proposed, an that as it has been seen, are all capable to calculate the damage probability. The criteria for choosing one or another is still to be studied.

# 8 Future Work

As it was said early, this is relatively a new line of investigation and thus little has been done, and much is still open for discussion.

One of the things that could be discussed is the effect of the dimension, i.e. the number of uncertain intervention times. In higher dimensions, Monte Carlo algorithms would probably prove better than the others, but still, which one of these algorithms is better. Another question that arises is which is the approximate number of dimension when it is better to switch to, or from, Monte Carlo techniques, and which one to choose.

An obvious line of investigation, is to release the convex domain restriction and try to compute the damage probability in the case of more complex domains. In practical examples,the damage domain are often not convex. But as most algorithms do not depend on a convex domain, the work will be centred in finding not convex models and to applied these exact algorithms to them. Still, real nuclear plant models are still not an option at these phase of the study as they are expensive in means of computational time.

Moreover, in the case of convex damage domain, is still to be studied if the Damage Time Interpolation Scheme can be modified to produce better results. As it was mentioned earlier, the results used in this work depended on the function to choose the number of simulations required and in the binary search mechanism. It may prove worthy to study possible improvements of these methodology.

# References


[1] J. M. Izquierdo. E. Melendez J. Devooght. *Relationships between probabilistic dynamics and event trees. Reliability Engineering and Systems Safety*, volume 52, page 197–209. 1996.

[2] J. M. Izquierdo et al. SCAIS (Simulation Code System for Integrated Safety Assessment): Current status and applications. In *Proc ESREL 2008, Valencia, Spain*, 2008.

[3] SMAP Task Group. Safety Margins Action Plan. Final report. Technical report, Nuclear Energy Agency. Committee on the Safety of Nuclear Installations.

[4] Javier Hortal. Schematic Example for Comparison of Uncertainty Analysis Methods, 2010.

[5] J. Hortal J. M. Izquierdo. *Application of the integrated safety assessment methodology to the protection of electric systems. Reliability Engineering and Systems Safety*, volume 52, page 315–326. 1996.

[6] J. M. Izquierdo. An integrated PSA approach to independent regulatory evaluations of nuclear safety assessment of Spanish nuclear power stations. In *EUROSAFE Forum, Paris, 2003*, 2003.

[7] J. M. Izquierdo and I. Cañamón. TSD, a SCAIS suitable variant of the SDTPD. In *Proc ESREL 2008, Valencia, Spain*, 2008.

[8] Izquierdo J.M. Sanchez M. Hortal J. and Melendez E. New techniques for the application of dynamic event trees to the interaction between electric grids and generating stations. In *Joint ESREL96-PSAM III conference, Creta, Greece 24-28 June 1996*, 1996.

[9] J.M. Izquierdo. J. Hortal E. Melendez I. Fernandez J. Gil S. Murcia J. Gomez C. Queral A. Exposito G. Rodriguez L. Ibañez M. Sanchez. Proposal for a suitable strategy of exceedance frequency computation. Implementation on SCAIS simulation-based safety code cluster. In *Proc. Int. Conf. Nuclear Energy for New Europe NENE2009,Bled, Slovenia, September 14-16, 2009*, page 140–154, 2009.

[10] A. Jourdain and P.E. Labeau. A monte carlo algorithm for dynamic psa based on the concept of stimulus. In *ANS PSA 2011 International Topical Meeting on Probabilistic Safety Assessment and Analysis*. American Nuclear Society, LaGrange Park, IL, 2011.

[11] P.E. Labeau and J.M. Izquierdo. *Modelling PSA problems I: The stimulus-driven theory of probabilistic dynamics. NSE Nuclear Science and Engineering*, volume 150, page 115–139. 2005.

[12] P.E. Labeau and J.M. Izquierdo. *Modelling PSA problems II: A cell-to-cell transport theory approach. NSE Nuclear Science and Engineering*, volume 150, page 140–154. 2005.

[13] Nicolas De los Santos Cicutto. Risk Assessment of a water tank protection system using SCAIS, 2011.

[14] Hortal J. Melendez E. Izquierdo J.M. and Sanchez M. *Automatic generation of dynamic event trees: a tool for integrated safety assessment (ISA). Reliability and Safety Assessment of Dynamic Process Systems*, volume 120. Springer Verlag, 1994.

[15] Hortal J. Melendez E. Izquierdo J.M. and Sanchez M. An integrated PSA approach to independent regulatory evaluations of nuclear safety assessment of Spanish nuclear power stations. In *Presented at the IBC 7th conference on PSA in the Nuclear Industry, Café Royal, London 26-27 November, 2001.*, 2001.

[16] Hortal J. Melendez E. Izquierdo J.M. and Sanchez M. An integrated PSA approach to independent regulatory evaluations of nuclear safety assessment of Spanish nuclear power stations. In *CSN, Colección Otros Documentos CSN, 28.2002, Madrid, 2002.* 2002.